Automated Bloodstain Segmentation Source Code Documentation

Analysis

bloodstain.py

This file defines the Stain object for the program and implementations of its methods. It also attempts to fit the stains into ellipses. Contains the contour and ellipse and statistics for each stain.

- __Init__(self, id, contour, scale, image)

Takes in id, contour, scale and image. Position and area are calculated using contour. Area_mm is calculated using the area and the scale. Calculated ellipse and contour line are returned by calling the fit_ellipse method. Major axis is initialised to none. The initialisation also checks if ellipse is not set to none, and if it is not none, x_ellipse, y_ellipse, width, height and angle are set to ellipse. If ellipse is none, x, y, width, height and angle are set to none.

- fit_ellipse(self)

This function attempts to fit the bloodstain into an ellipse to calculate the width, length, x and y coordinates of the centre and the angle. It will only calculate if the contour array length is bigger than or equal to 5 and the contour area is bigger than 9. Before the fitting the ellipse, the tail-cutting process is carried out. This is done by identifying the concave points along the bloodstain boundary. Then starting from the furthest point away from the centroid, the algorithm steps around the stains in both directions measuring the distance between the two points. When the ratio of this distance to the maximum width or height is greater than 0.5 and one of the points is a concave point then it is the end of the tail. All points between these two points are removed.

- draw_ellipse(self, image)

cv2.ellipse() is used to draw an ellipse on an image.

- circularity(self)

This method calculates the circularity of the ellipse if it exists by getting the ratio of width and height. Value close to 0 would mean more elliptical, closer to 1 would mean more circular.

- orientation(self)

This method calls the direction method to get what direction the ellipse is in and calculates the gamma value and returns it alongside with the angle. If the ellipse does not exist, the values return as infinity.

- direction(self)

This method determines the general direction of the ellipse by using angle and contour. The first value indicates the x axis direction and the second value indicates the y axis direction. It first creates a convex hull of the contour line. And then it gets the difference of between the position coordinate of the contour line and each point in the convex hull. The x and y values of the differences are stored in a list. The x and y values are squared and combined, and then square rooted. The point with the

highest value is chosen to determine the direction. The point is then combined with the position coordinate of the contour. If the x value is lower than 0, the x axis direction is determined left, otherwise right. If the y value is lower than 0, the y axis direction is determined up, otherwise down.

- calculate_major_axis(self)

this method calculates the major axis for the ellipse using the direction given by the direction method and the most left, right, up and down points.

- area_half(self, half_contour)

This method determines the half area of the ellipse using the half contour given if the length of the half contour is bigger than 0. This method is deprecated.

- intensity(self)

This method determines the average colour intensity of the pixels contained within the bloodstain. The value returned is between 0 and 1.

- solidity(self)

This method determines the regularity of the element margin by getting the ratio of the area of the ellipse and the convex hull area of the contour. The value returned is between 0 and 1.

- annotate(self, image, annotations)

This method annotates the ellipse, id number, directionality, the centre point of the ellipse, the gamma value, and the direction line onto the image. Each feature can be turned be on and off by making changes to the annotations parameter. The annotations parameter will default to ellipse to true, id to false, directionality to false, centre to false, gamma to false, and directional line to false.

- label(self)

This method returns an array containing the id of the ellipse and the contour points.

- obj_format(self, width, height)

This method returns a string containing each point of the contour divided by width and height.

- get_summary_data(self)

This method takes the current data on the ellipse and reformats the data and returns it.

- write_data(self, writer)

This method uses the writer and the get_summary_data to record the data.

pattern.py

This file defines the pattern object and implementations of its methods.

- __init__(self, image, thresh, filename, scale)

This method initialises the parameters to its properties. It also initialises other properties such as contours, stains, elliptical_stains, summary_data and plots. The scale parameter defaults to 7.0.

- add_stain(self, stain)

This method adds a new stain to the stains array. If the stain has a major axis, it also gets added to the elliptical_stains array.

- convergence(self)

this method finds the intersections between bloodstains and returns the returned value of plot_convergence method using the intersections as the parameter.

- plot_convergence(self, intersects, figsize)

This method creates a new figure and use other methods such as plot_intersection_scatter, calculates_convergence_box, and plot_density_heatmap to plot the convergences on the figure. It also creates a new window containing convergence chart.

- plot_intersection_scatter(self, ax1, x, y)

This method plots a scatter plot of directional major axis intersections on the figure.

- plot_density_heatmap(self, point_density, xi, yi)

This method plots a heat map of convergence on the figure.

- calculate_convergence_box(self, point_density, xi, yi)

This method calculates the convergence box using the densest area and the convergence point

- line_intersection(self, line1, line2)

This method calculates the point where two lines intersect. If the two lines do not intersect, the method returns none.

- linearity(self, figsize)

This method that fits the stain centres to a degree 2 polynomial. The figsize parameter defaults to (18, 12) if not set.

- distribution(self, figsize)

This method calculates the convex hull of the stains and the ratio of the convex hull to the number of stains and the area occupied by stains to the volume of the hull. The figsize parameter defaults to (18, 12) if not set.

- calculate_summary_data(self, pattern_metrics)

This method calculates and summarises the data by calling the methods in the class and putting them in a list.

- get_summary_data(self, pattern_metrics)

This method returns the data summarised by calculate_summary_data method if the length of the list of the summarised data is longer than 0.

- export(self, summary_data, save_path)

This method saves the summarised data in a csv file in the save path specified. With the column headers being, ‘Linearity – Polyline fit’, ‘R^2’, ‘Distribution – ratio stain number to convex hull area’,

‘ratio stain area to convex hull area’, ‘convergence – point of highest density’, and ‘box of %60 of intersections’.

stain_segmentation.py

This file contains functions for image processing and drawing on the input image. It uses the bloodstain and pattern files.

- process_image(filename, output_path, scale, show, pattern_metrics)

This function loads the image and does stain segmentation and fetches the summarised data by calling various other functions in this file and in other files. The scale parameter defaults to 7.0 if not set. The show parameter defaults to False. The pattern_metrics parameter defaults to None.

- draw_stains(pattern)

This function annotates the stains on the image.

- export_pattern(pattern, stain_overlay)

This function exports the pattern into the specified path and saves the pattern as an image. It also exports stain data and the pattern object in the same path.

- find_images(folder, file_types)

This function finds all image files in a folder and returns a list of the names of the image files in the folder. The file_types parameter defaults to a list of file types including jpg, jpeg, gif, png, tif, and tiff files if not set.

- batch_process(input_path, output_path, scale, show, overwite, pattern_metrics)

This method uses the image files found with the find_images functions to process each file found in a folder. The output_path defaults to None, scale to 7.0, show to False, overwrite to False, and pattern_metrics to None.

- stain_segmentation(image, filename, scale)

This function defines the pattern for the image and returns the pattern object. It also finds the contours and analyse them. The scale parameter defaults to 7.0 if not set.

- analyseContours(pattern, contours, hierarachy, image, scale)

This function analyses the contours given and updates the number of the stains in the pattern.

- export_stain_data(save_path, pattern)

This function exports the stain data in 2 different csv files, ‘data.csv’ and ‘stains.csv’. The data file contains the data about each stain. Information such as the id, position x, y, area px, area mm, width ellipse, height ellipse, angle, gamma, direction, solidarity, circularity, and intensity. In the stains file, the labels for the stains are saved.

- export_obj(save_path, pattern)

This function records a pattern object into the points.pts file in the save_path parameter.

- result_preview(img_original)

This function displays the smaller version of the original image until the user presses the Q button on the keyboard and destroys all windows on the screen.

- remove_circle_makers(grey, img)

This function finds the circles in an image and draws the circles on the image.

- binarize_image(gray)

This function finds the erosion and dilation of the image and flips the bits in dilation.

- label_stain(pattern)

This method is used to label stains and exports the them in a json file.

- show_intensity_histogram(img)

This function displays the histogram of the intensity of the image.

App

main.py

This file contains the functions needed to set up the application, including setting up the GUI window that lets the user load up the image. It also contains the BPA_App class that does the basic operation for the application.

- image_to_pixmap(image)

This function converts an image file to a pixmap onject.

- select_export_dialog(filename)

This function opens or creates a directory if it doesn't exist already.

- main()

This method runs the functions needed for the application to initalise.

BPA_App(QtWidgets, QMainWindow, main_window.Ui_MainWindow)

- __init__(self, parent)

This method initialises the application. This function also sets up the UI and creates the main window for the application which includes the photo viewer.

- load_image(self)

This method loads the image file from the folder.

- open_image(self, file_name)

This method checks that the image file exists then opens the image.

- export(self)

This method checks that the export path exists and exports the pattern into the selected path.

- show_metric(self)

This method shows the metrics on the window if there is an image file loaded. Otherwise displays a message box.

- show_dialog(self, dialog, accept)

This method sets up the metrics for display.

- update_scale(self, value)

This method updates the scale to the value given.

- segment_image(self)

This method segments the image using the functions from the stain_segmentation file. It then displays the pattern image from the output and populates the tables.

- populate_tables(self)

This method clears then populates the stain table and the pattern table.

- populate_stain_table(self)

This method populates the stain table with data from the stain list.

- show_stain(self, item)

This method places a rectangle on the stain.

- populate_pattern_table(self,)

This method populates the pattern table using the data from the pattern summarised data.

- clear_tables(self)

This method clears the data in the tables.

- show_batch_dialog(self)

This method shows the metrics from the batch processing.

- open_folder(self)

This method sets the open folder chosen by the user,

- output_folder(self)

This method sets the output folder chosen by the user.

- batch_process(self)

This method processes the image files in a batch.

photo_viewer.py

This file contains the photo viewer object and its methods. This object is used to display images in the application window.

- __init__(self, parent)

This method initialises the photo viewer by setting up its properties.

- hasPhoto(self)

This method returns a Boolean value whether or not if the photo viewer has an image in it.

- fitInView(self, scale)

This method transforms the image to fit in the photo viewer. The parameter scale defaults to True if not set.

- setPhoto(self, pixmap)

This method sets the given pixmap as an image in the photo viewer and fits in the view.

- wheelEvent(self, event)

This method sets the wheel event for the event.

- toggleDragMode(self)

This method sets the drag mode if not set. If the photo pixmap is not set, it sets the scroll hand drag.

- mousePressEvent(self, event)

This method handles the mouse press events, passing on the mouse position.

- add_rectangle(self, x, y, width, height)

This method adds a rectangle to the image in the given position.

- set_text(self, x, y, text)

This method adds text to the stain that has been clicked on.

- add_text(self, stain, text)

This method adds text for the stain on the image.

- add_outline(self, stain)

This method adds an outline on the stain on the image.

- add_direction_line(self, stain)

This method adds a direction of the stain on the image.

- add_center(self, stain)

This method adds the centre point of the stain on the image.

- add_ellipse(self, stain)

This method puts an ellipse around the stain on the image.

- add_annotations(self, annotations, pattern)

This method adds annotations given on the image.

Generated

batch_process.py

This file contains the class UI batch processing and its methods. UI_BatchProcessing is a subclass of object superclass.

UI_BatchProcessing(object)

- setupUi(self, batchProcessing)

This method sets up various UI elements in the window for the batch processing such as buttons and labels.

- retranslateUi(self, BatchProcessing)

This method retranslates the UI elements and sets texts.

features_dialog.py

This file contains the class UI segmentation metrics class and its methods. Ui_SegmentationMetrics is a subclass of object superclass.

- setupUi(self, SegmentationMetrics)

This method sets up various UI elements in the window for the features dialog such as label and lines.

- retranslateUi(self, SegmentationMetrics)

This method retranslates the UI elements and sets texts.

main_window.py

This file contains the class UI main window and its methods. Ui_MainWindow is a subclass of object superclass.

Ui_MainWindow(object)

- setupUi(self, MainWindow)

This method sets up various UI elements for the main window such as widgets and the menu bar.

- restanslateUi(self, MainWindow)

This method retranslates the UI elements and sets texts.

processing.py

This file contains the class UI Dialog and its methods. Ui_Dialog is a subclass of object superclass.

UI_Dialog(object)

- setupUi(self, Dialog)

This method sets up various UI elements in the window for the dialogs such as labels.

- retranslateUi(self, Dialog)

This method retranslates the UI elements and sets texts.

Scripts

analysis.py

This file contains the arguments for the parser and the analysis process script.

- parse_args()

This function adds arguments to the parser for the stain segmentation and returns the parser.

- main()

This function contains the script for the analysis process.

crop.py

This file contains the functions for cropping the image.

- remove_rulers(image)

This function crops the ruler out of the image. It only works with clear rulers on all sides.

- line_count(x1, x_count)

This function counts the number of lines that start at the same point.

- remove_bottom(no_ruler_crop, y)

This function is used to remove the bottom of images.